exlepse

Eclipse XLE Plugin

Motivation

During the course „Grammar Development“, held at the University of Konstanz in the Summer Term 2010, we were confronted with XLE and its development environment, the emacs editor. The nonconformity of the emacs editor with common editor concepts proved to be a major obstacle for the development of grammars, especially for novices in this field. Having a background in HCI, we decided to develop a XLE Editor, which better supports users trying to implement and test a grammar in XLE.

Our goal is to lower the learning curve for any XLE novice, by letting them concentrate solely on the development of the grammar without painfully learning the details of a new editor and its concepts.

There are not only problems with the emacs editor which we like to address with our tool. Things like bad error handling in XLE or the syntax of XLE, which isn't very intuitive, are also topics we think our tool could support better.

A further point addresses the organization of bigger grammar projects. Grammars in those projects are comprised of several smaller units, which are managed with some kind of version management system (like Subversion). There's no built in support for such tools in the emacs editor, however.

The problems which were only touched here shortly are listed below in more detail and the design and usage of our tool is explained.

Problems in grammar development with XLE

Being very sensitive concerning usability problems in user interfaces, the following problems immediately struck our attention:

Cryptic Keyboard Shortcuts

The emacs editor uses default keyboard shortcuts that conform to no current convention. This complicates the fluent grammar development, as the user needs to shift focus away from his grammar problem to a usability problem.

Sentence parsing

The parsing of sentences in the emacs editor takes place in a separate buffer in addition to the grammar and testsuite buffer. There's the possibility to place two buffers on top of or next to each other which must be configured separately each time. However whenever the developer likes to parse a sentence, he is forced to switch his current context to another buffer, which might not be visible at the moment.

XLE Fehlermeldungen

XLE prints error messages to the standard console. However the details of these error messages are problematic. Most often only sparse clues are given to identify the error in the grammar. Even worse error messages are printed into an extra buffer which forces the developer to switch buffers in order to view the error in the context of its grammar.

XLE Syntax

The syntax of the various XLE parts (config, rules, templates) in some cases looks very arbitrary. In combination with the mediocre error messages this can lead to frustration.

An example shall be given here:

Delimiters

The delimiter which separates single parts (e.g. lexicon entries) in XLE is a dot. The dot is the smallest visible lexical symbol and therefore hard to find on the screen. Many errors in grammar development are the result of a missing dot or other delimiters (e.g. semicolons).

While developing our editor we noticed that using a dot as delimiter furthermore causes problem with parsing. In many languages the dot is the finishing symbol of a sentence. In grammars for these languages the dot therefore also appears in lexicon entries. This complicates parsing of the grammar extremely, since both semantics of the dot must be considered.

Technical Implementation

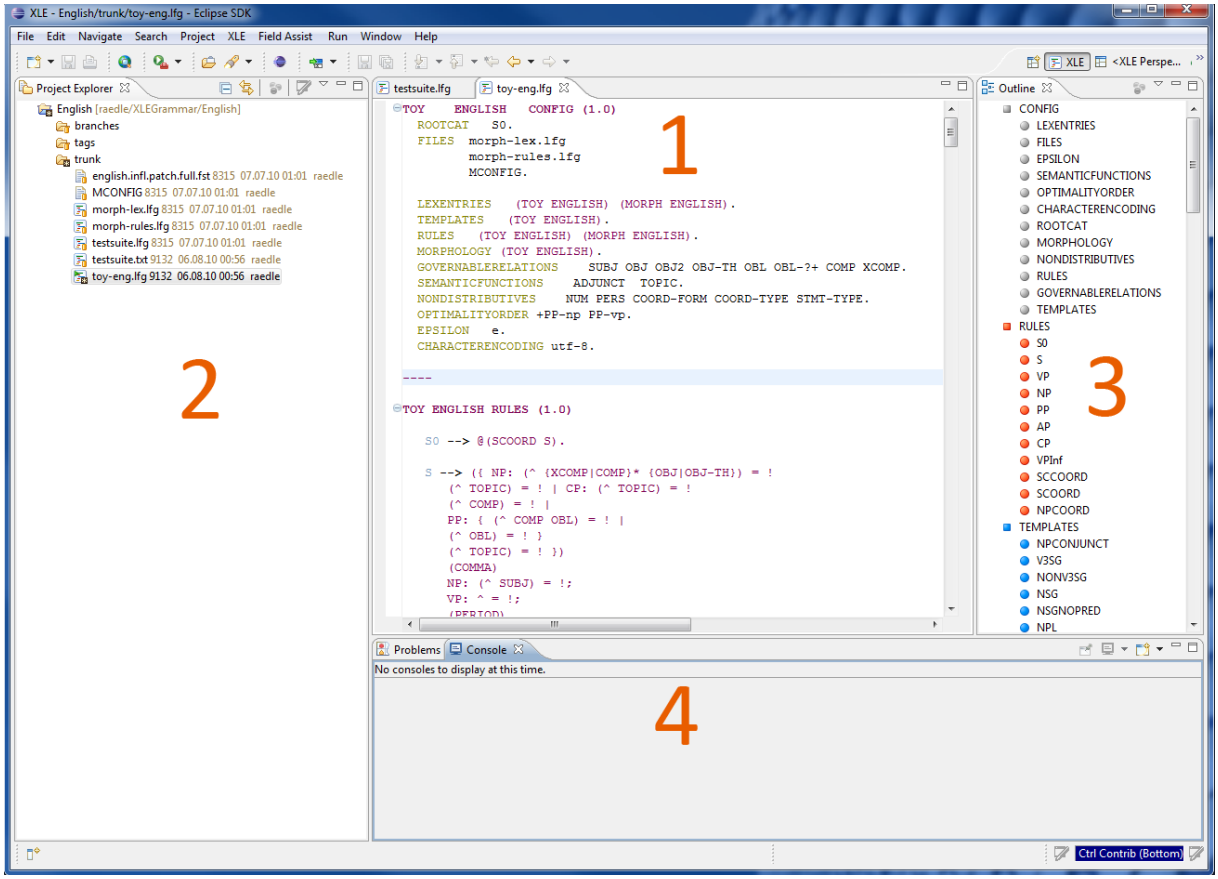
Eclipse Plug-In

Eclipse is a powerful open-source development platform, which is used mainly in software development. Eclipse is meant to be extended by so called plug-ins in various ways. Our decision to do an Eclipse plug-in was based on the fact, that many components of Eclipse were ready to use, letting us concentrate more intensively on the implementation of XLE-specific components.

As mentioned before, our tool is a plug-in for the standard Eclipse version. It can be installed via the default update mechanism of Eclipse (see section **Installation**).

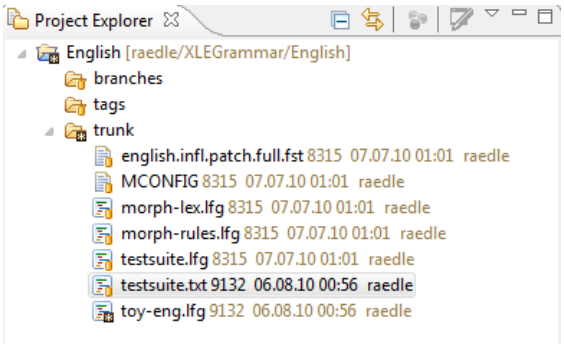
The XLE perspective

The first thing you'll notice when installing the plug-in, is that we customized Eclipse in a way that only the components needed for XLE grammar development are visible. This XLE perspective consists of one or more editors in the central work area (1), a project explorer (2) on the left, an outline (3) on the right side and a complete XLE console as well as an error view (4) on the bottom.



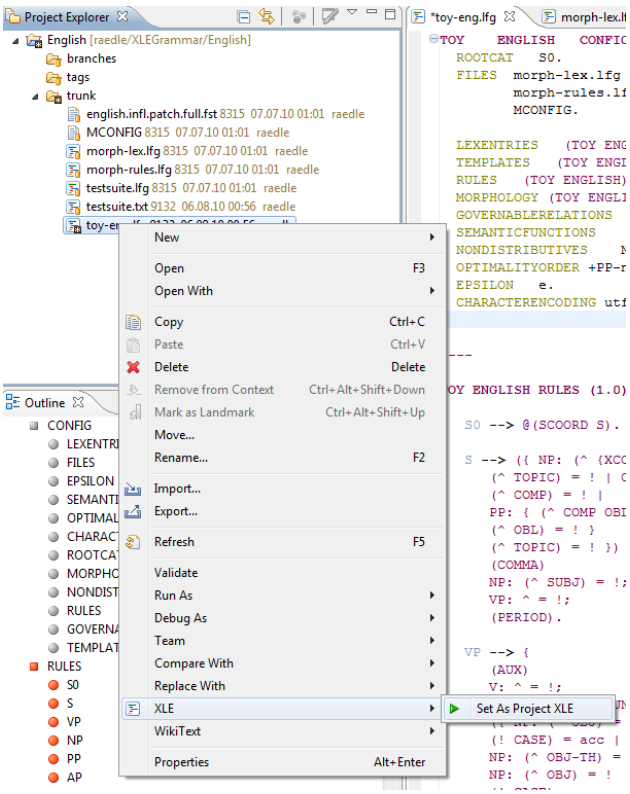
Project Explorer

The project explorer serves as an overview of all files existing in a grammar project. Arbitrary files or folders can be created or managed here. If a version management system (see below) for distributed development is used, the checked out project is shown here.



Choosing the main file

In order to parse sentences our tool needs to know which file of the project is the starting point for parsing. This can be achieved by calling the context menu on the particular file in the project explorer:

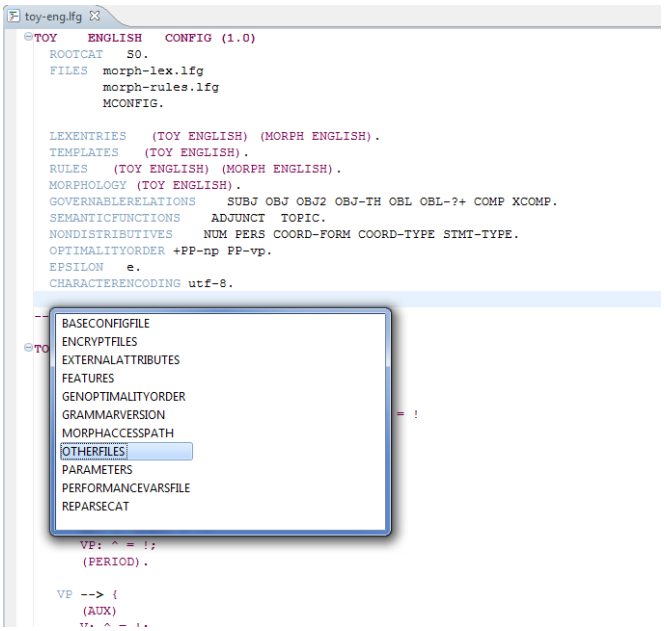


LFG Editor

The LFG editor is the central and most important part of our plug-in. It is used to create and edit all the files of a grammar project. Its functionalities are presented below:

Content Assist

In the config section of a LFG-File the user can request proposals for editing (via Ctrl+Space or Edit -> Content Assist in the menu). The proposals consist of all config entries that are not used in the current file.

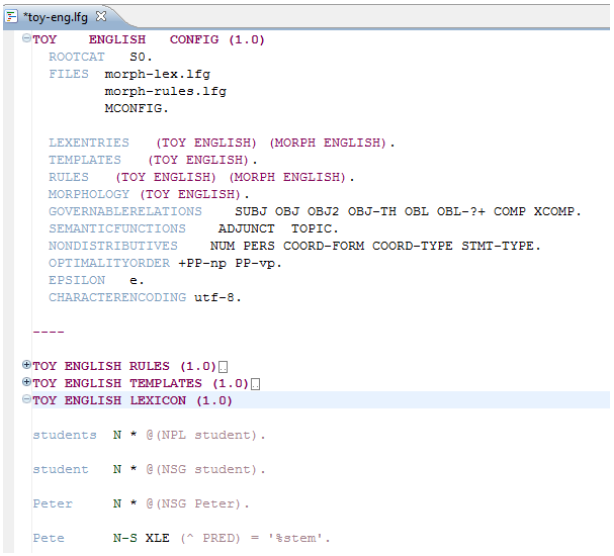


Syntax Highlighting

In all parts of a LFG-File (CONFIG, RULES, LEXICON, TEMPLATES) the text is colored individually. This syntax highlighting improves readability of complex expressions. The colors of each part can be changed separately in a preference dialog (see section **Preferences**).

Folding

Particular parts of a LFG-File can be folded with the help of a little symbol on the left margin of the editor. This leads to a better overview especially of large grammar files.

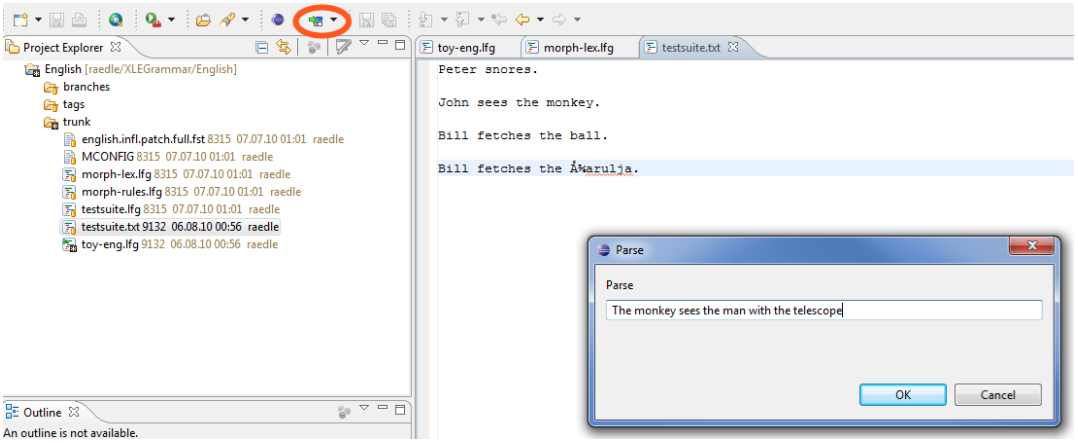


Encoding (UTF-8)

The LFG editor by default uses the UTF-8 Unicode encoding, so that characters of slawic languages for example can be used correctly. If a different encoding is needed, it can be changed in the preferences dialog.

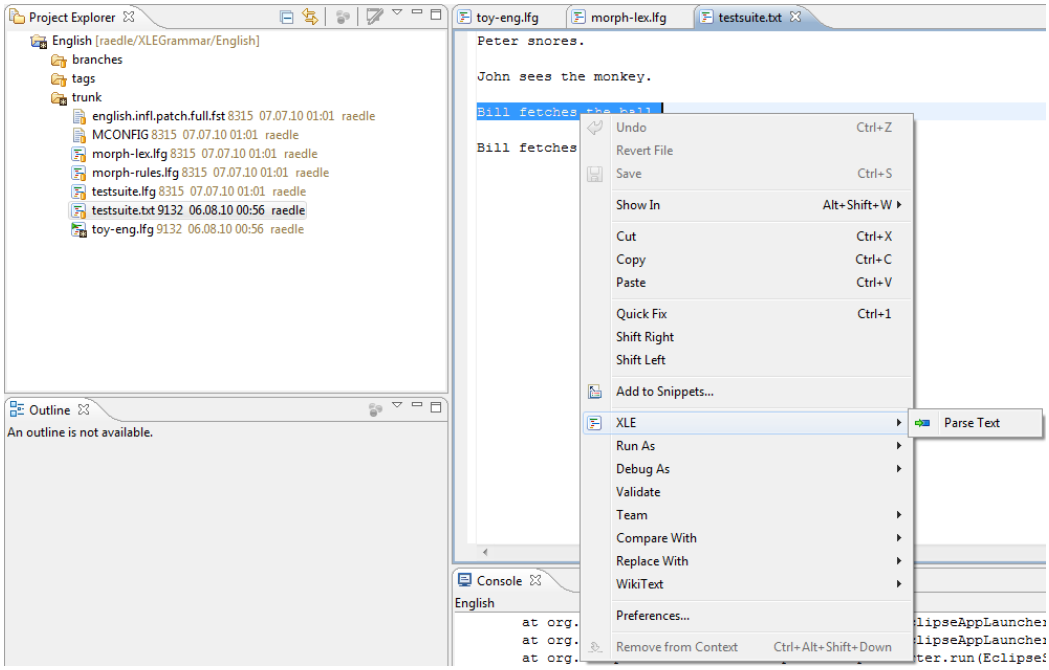
Parsing

An important point for efficiently developing a grammar is quick parsing of sentences or fragments of sentences. If no test suite is used the developer can request a dialog via the toolbar. Text that is given as input here is directly sent to the XLE parser.



Testsuite Editor

If you want to use test suites they can easily be loaded into their separate editor. In this editor, parts of the test suite can be sent to the XLE parser by selecting those parts and calling the appropriate command of the context menu or toolbar.

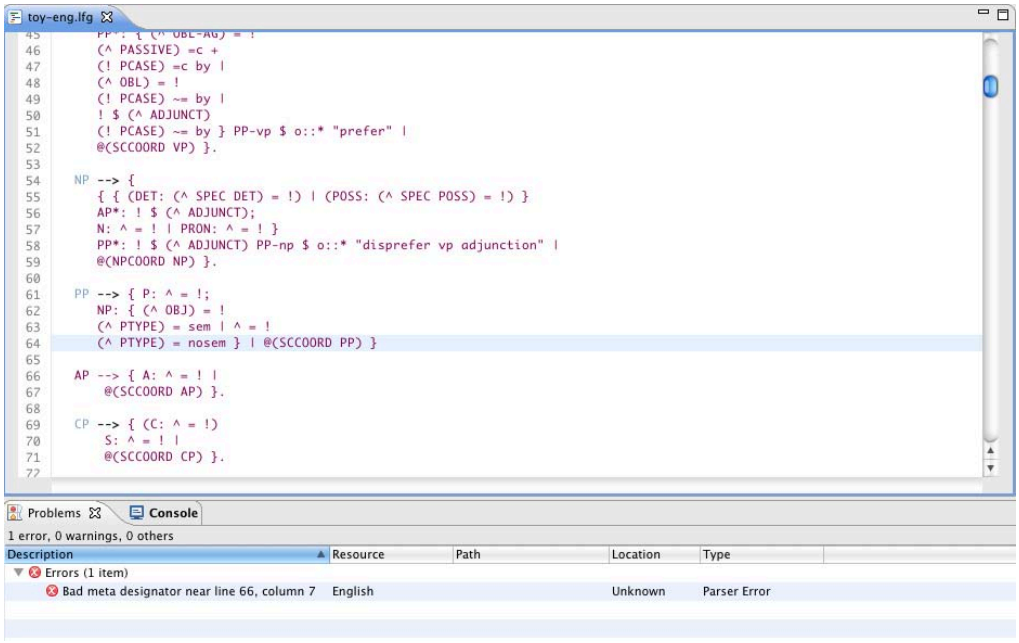


Console Output

At the bottom a full-fledged XLE console is opened after a call to the parser. Results of parsing are printed here, but there's also the possibility for the developer to enter commands which are interpreted by the XLE parser then. It's therefore possible to manually parse sentences (by using the parse "my sentence" command). Furthermore it's possible to terminate the current XLE process which kills all available XLE windows.

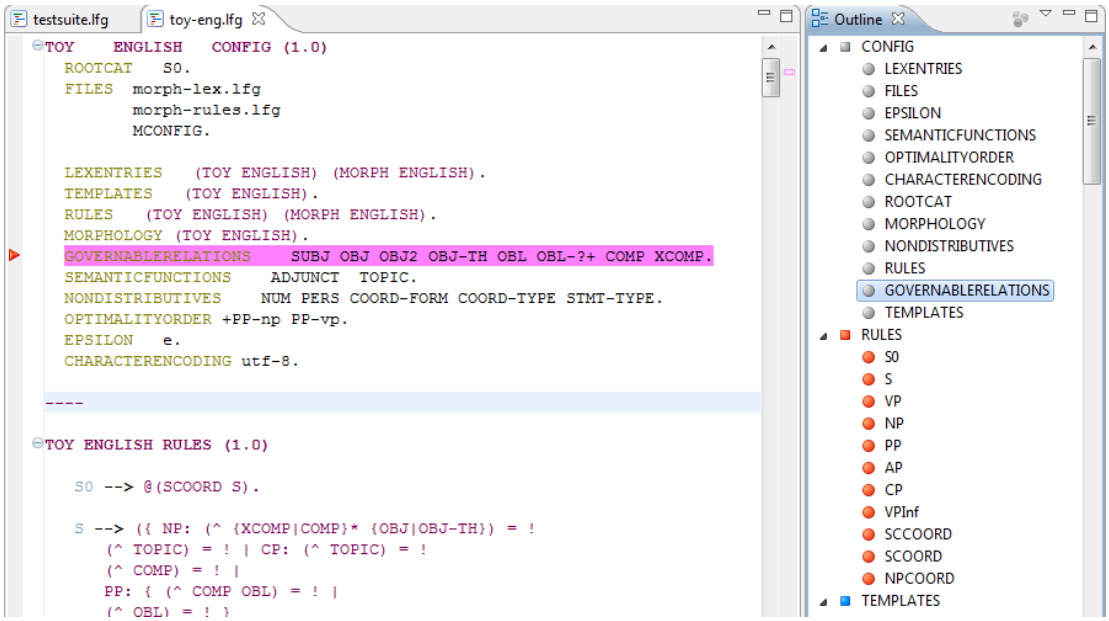
Fehlerbehandlung

Errors that occur during parsing are shown in a special view at the bottom of the window. The developer immediately gets feedback about the location of the error and can navigate directly to it.



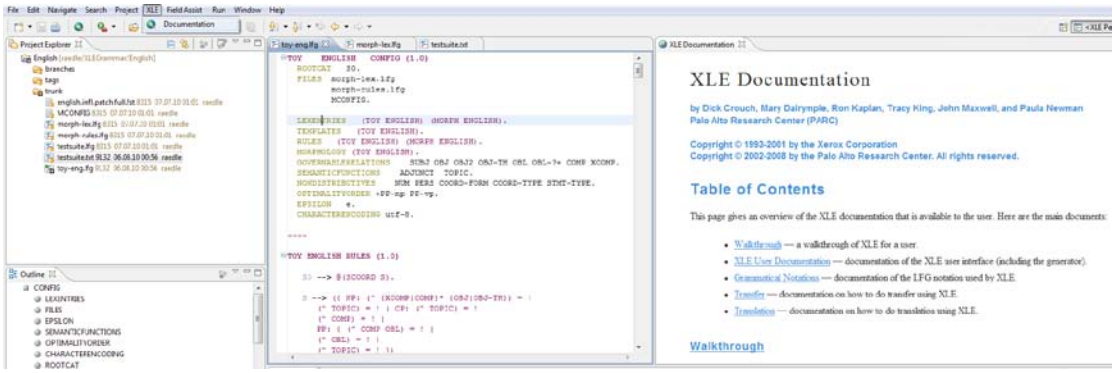
Outline

On the right side of the window a quick outline of the currently opened LFG-File is shown. This outline consists of all parts of an LFG-File and its main entries (rules, lexicon entries, template definitions). Selecting one of these entries in the outline view causes the editor to place a marker on the corresponding line and color that line differently.



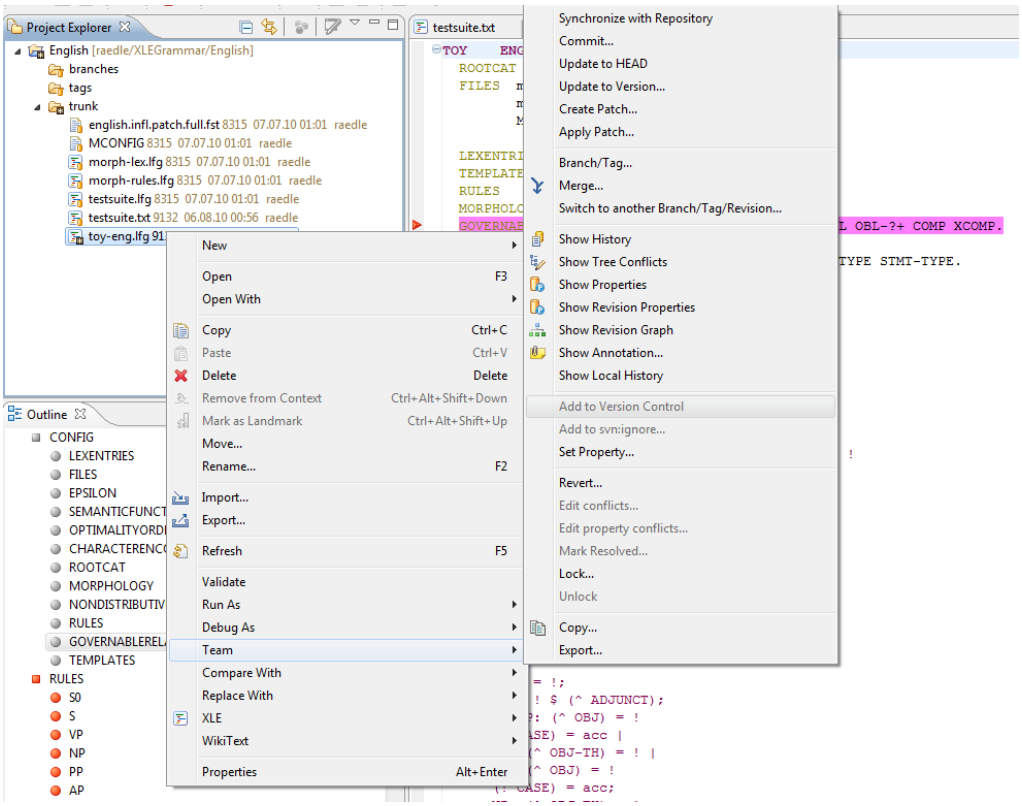
XLE Documentation

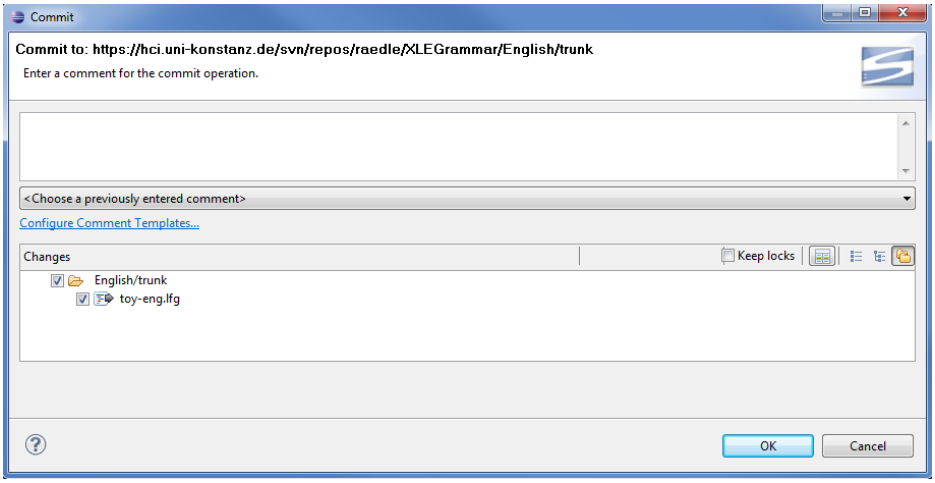
The official XLE documentation can be shown in a separate editor window in parallel to other opened editors. It can be reached via the menu or toolbar.



Subversion Integration

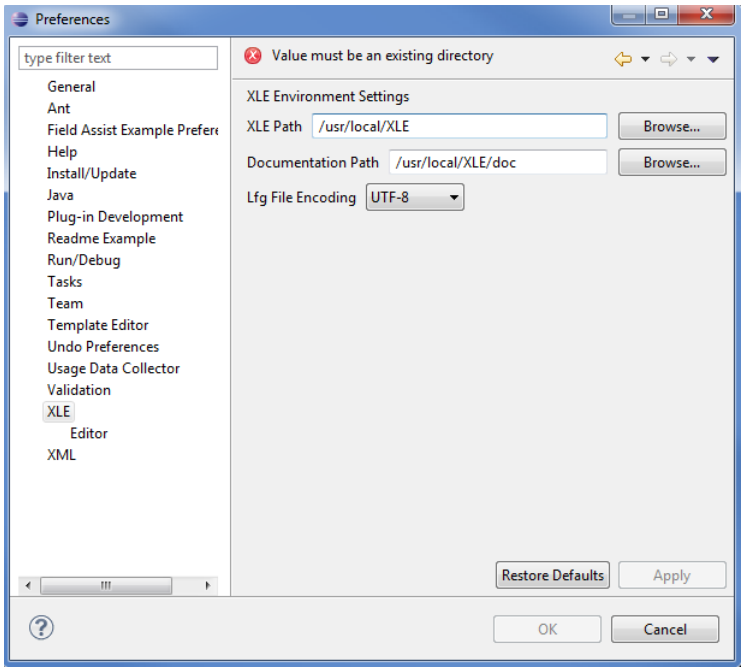
Bigger, distributed grammar projects use version management software like Subversion for the organization of their project sources. There's built-in support for subversion (and other versioning software) in Eclipse via a plug-in (subclipse). Thus a developer can use all features of subversion without leaving Eclipse.



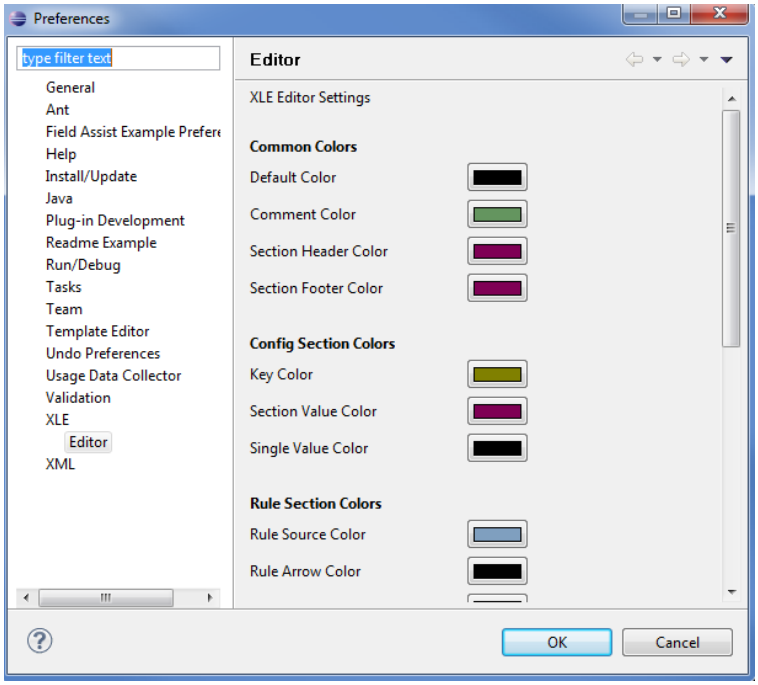


Preferences

Some parameter of our plug-in can be adjusted via the preference dialog of Eclipse. The paths to the XLE binaries and the documentation or the default encoding for LFG-Files can be set for example.



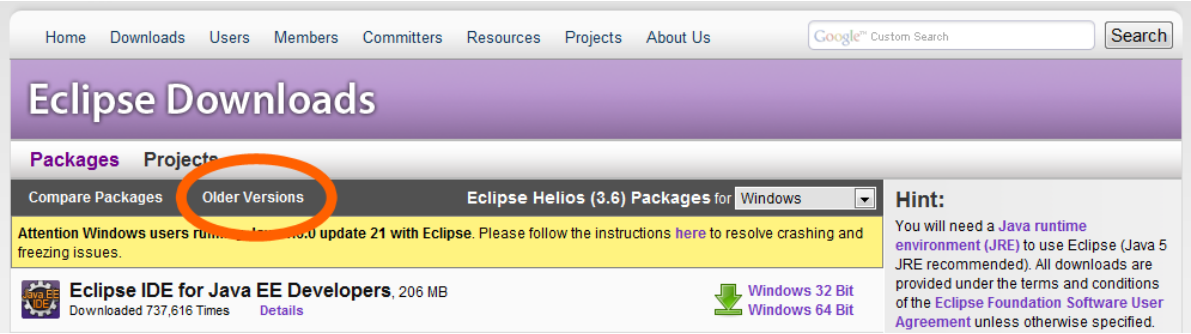
A further dialog can be used to set the colors of the syntax highlighting of the editor.

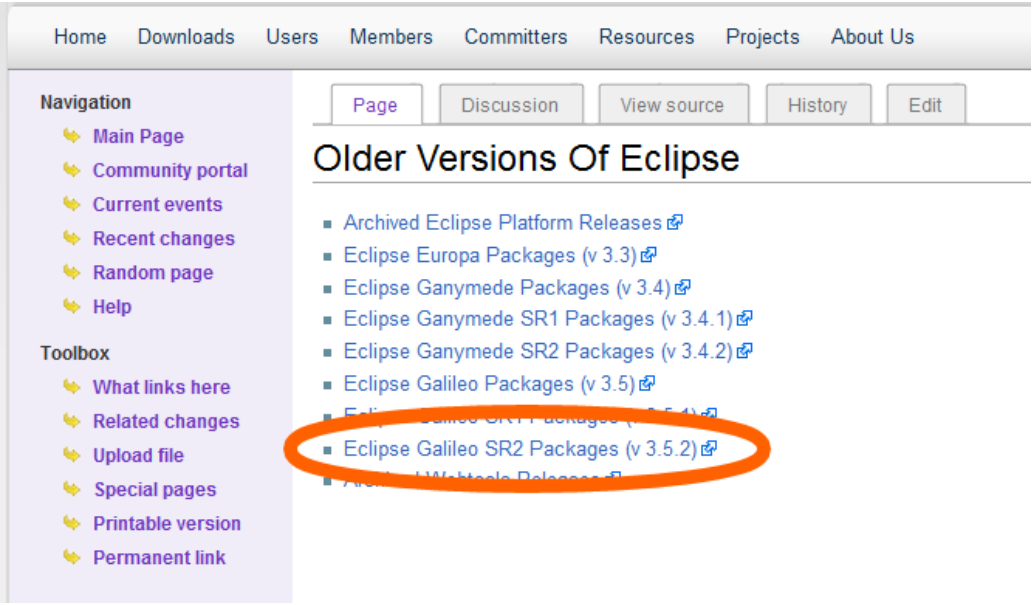


Installation

Our plug-in can be installed using the default installation mechanism of Eclipse. What follows is a step-by-step instruction of the necessary actions:

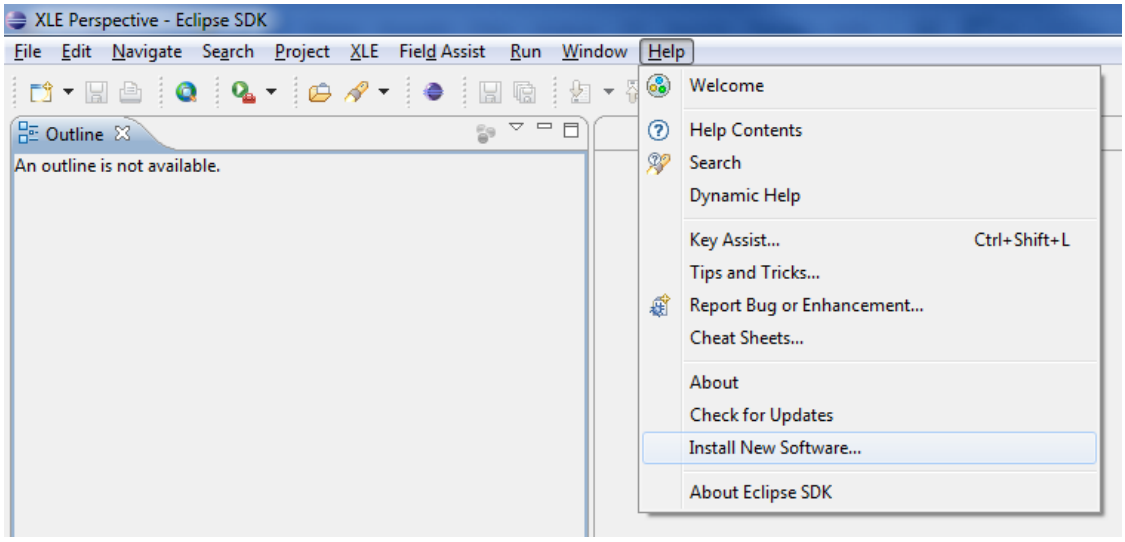
1. ***Download Eclipse***. The website http://www.eclipse.org/downloads/ offers various versions of Eclipse for downloading. For our plug-in Eclipse Classic Version 3.5.2 is needed:



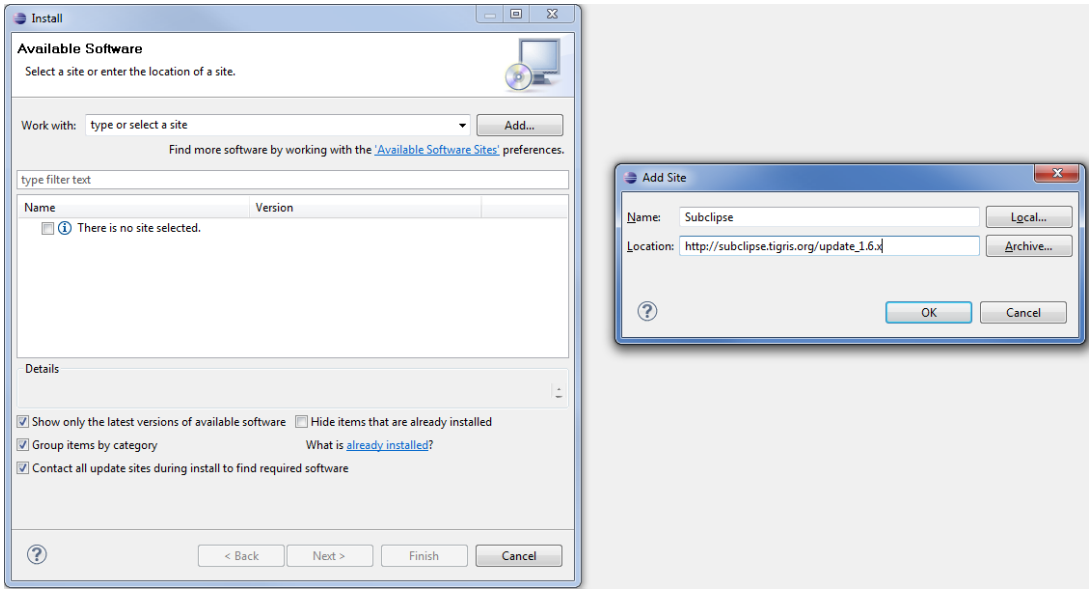


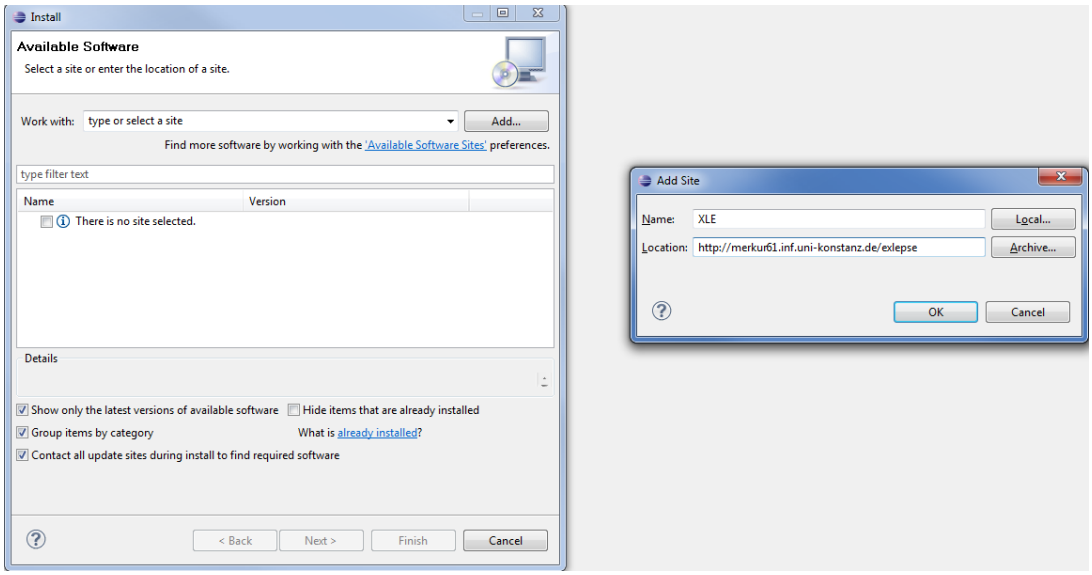
After the download of eclipse has finished, it can be extracted to any folder on your hard drive.

2. ***Installing the plug-in***. Select the command Help → Install New Software in the menu.

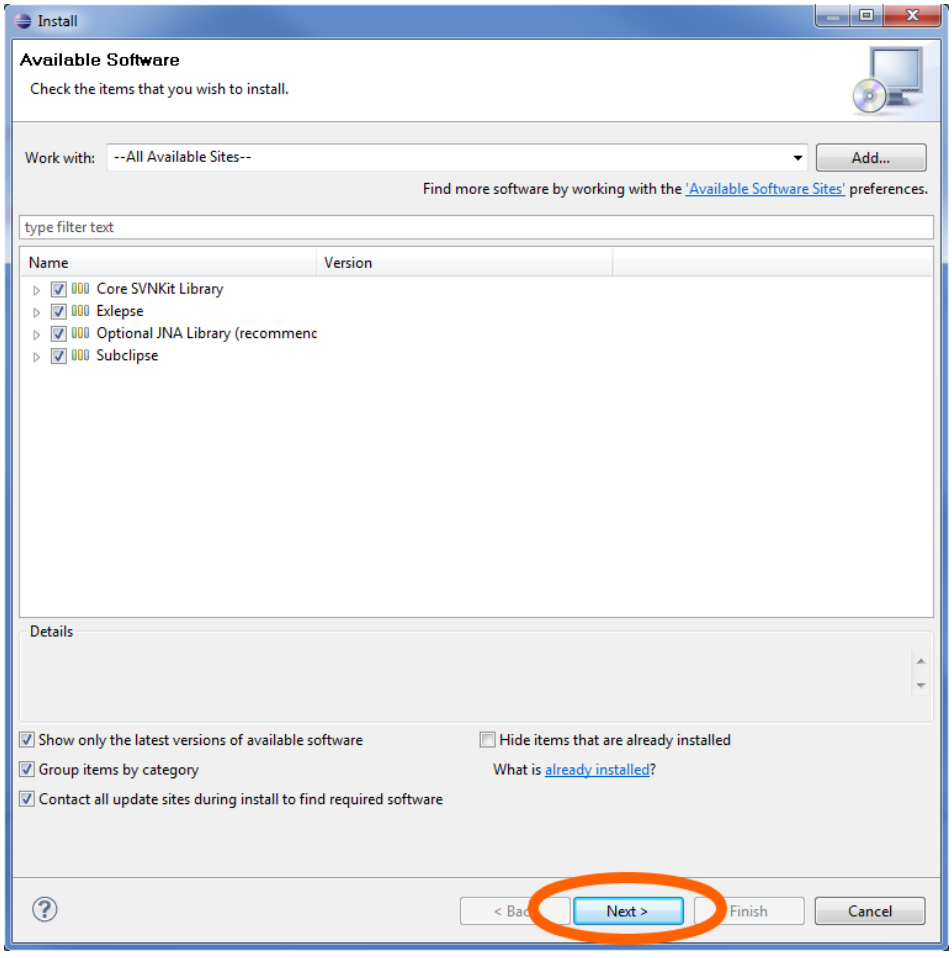


Using the Add-Button you first need to enter the installation site of Subclipse (the subversion plug-in), then the installation site of our plug-in (http://subclipse.tigris.org/update_1.6.x and http://merkur61.inf.uni-konstanz.de/exlepse).





Next check all entries that are available and complete the dialog.



We already prepared a ready-to-use Eclipse-Bundle, which can be found at the following address: http://merkur61.inf.uni-konstanz.de/exlepse/exlepse-1.0.zip. All required plug-ins are already installed therein.